



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

12

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/266,675	03/11/1999	RANDY S. KIMMERLY	777.278US1	6126
41505	7590	03/17/2006	EXAMINER	
WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION) ONE LIBERTY PLACE - 46TH FLOOR PHILADELPHIA, PA 19103				LY, ANH
ART UNIT		PAPER NUMBER		
2162				

DATE MAILED: 03/17/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/266,675	KIMMERLY, RANDY S.	
	Examiner	Art Unit	
	Anh Ly	2162	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) Responsive to communication(s) filed on 19 December 2005.
- 2a) This action is FINAL. 2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) Claim(s) 1-24 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) Claim(s) _____ is/are allowed.
- 6) Claim(s) 1-24 is/are rejected.
- 7) Claim(s) _____ is/are objected to.
- 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on _____ is/are: a) accepted or b) objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) All b) Some * c) None of:
1. Certified copies of the priority documents have been received.
 2. Certified copies of the priority documents have been received in Application No. _____.
 3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413) |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | Paper No(s)/Mail Date. _____ |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____ | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. This Office Action is response to Applicant's response filed 12/19/2005.
2. Claims 1-24 are pending in this application.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No 5,187,786 issued to Densmore et al. (hereinafter Densmore) in view of Pub. No.: US 2001/0047510 A1 of Angel et al. (hereinafter Angel).

With respect to claim 1, Densmore teaches generating a cache of information relating to the classes in the class path (the implementing a root class with a class hierarchy of objects being a class path containing a plurality of class path directory names, and its elements are class files and class instance, since a path or directory is a hierarchical structure: abstract, col. 2, lines 46-67 and col. 3, lines 1-14 col. 4, lines 46-67 and col. 5, lines 1-28 and the root class and class instance directory are stored in the system: col. 4, lines 64-67); creating a wrapper (hierarchy of root class and class instance directory are a wrapper: col. col. 4, lines 60-67 and col. 5, lines 1-28); requesting a search of the class path and searching the cache to satisfy the requested search (search path: col. 9, lines 6-9 and lines 15-32).

Densmore teaches creating a root class with a class hierarchy of objects in a hierarchical file system, objects are organized as class instances and classes, the data are contained in the class variables and/or the class instance variables, a hierarchy of a root class directory and root class files, a class making procedure comprising an interface for receiving a class specification as well as for receiving a message (abstract, col. 2, lines 46-67, col. 3, lines 1-14, col. 4, lines 46-67 and col. 5, lines 1-28), setting the search path of hierarchical file system to the content of the class instance or class file under unchanged current directory (col. 9, lines 14-22). Densmore does not explicitly teach a level of indirection from application programming interfaces used by a class locator, the wrapper indirection level providing for different caches to be used for the selected elements.

However, Angel teaches creating a Java byte code wrapper for locating or searching finding or retrieving the class from the constant pool of the class instance and for providing a level of indirection from the application programming interfaces (APIs) for each query objects (sections 0016, 0126, 0132 0135 , 0161 0168; also see fig. 17).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Densmore with the teachings of Angel, wherein the interface for receiving or selecting the element such as a class specification as input in the system provided therein (Densmore's col. 4, lines 52-59 and col. 5, lines 5-28), would incorporate the use of creating a wrapper for selected object (Angel's section 0016 and 0168), into the system of Densmore for the

purpose of efficiently searching/locating the classes in the class path (Densmore's col. 9, lines 6-9 and lines 15-32).

Claim 2 is essentially the same as claim 1 except that it is directed to a computer readable medium rather than a method, and is rejected for the same reason as applied to the claim 1 hereinabove.

With respect to claim 3, Densmore wherein the class path comprises multiple elements, each element having multiple classes stored therein (class hierarchy of object: comprising root class, a plurality of classes: see abstract).

With respect to claim 4, Densmore teaches a method of locating classes as discussed in claim 1.

Densmore teaches creating a root class with a class hierarchy of objects in a hierarchical file system, objects are organized as class instances and classes, the data are contained in the class variables and/or the class instance variables, a hierarchy of a root class directory and root class files, a class making procedure comprising an interface for receiving a class specification as well as for receiving a message (abstract, col. 2, lines 46-67, col. 3, lines 1-14, col. 4, lines 46-67 and col. 5, lines 1-28), setting the search path of hierarchical file system to the content of the class instance or class file under unchanged current directory (col. 9, lines 14-22). Densmore does not explicitly teach a zip file.

However, Angel teaches zip file (section 0131).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Densmore with the

teachings of Angel, wherein the interface for receiving or selecting the element such as a class specification as input in the system provided therein (Densmore's col. 4, lines 52-59 and col. 5, lines 5-28), would incorporate the use of zip file from the VM runtime machine for selected object (Angel's section 0131), into the system of Densmore for the purpose of efficiently searching/locating the classes in the class path (Densmore's col. 9, lines 6-9 and lines 15-32).

With respect to claim 5, Densmore discloses generating a search request for desired classes within the multi-element class path; and independently satisfying the request in association with each element in the class path (abstract, col. 2, lines 46-67 and col. 3, lines 1-13; a wrapper (hierarchy of root class and class instance directory are a wrapper: col. col. 4, lines 60-67 and col. 5, lines 1-28); and search path: col. 9, lines 6-9 and lines 15-32).

Densmore teaches creating a root class with a class hierarchy of objects in a hierarchical file system, objects are organized as class instances and classes, the data are contained in the class variables and/or the class instance variables, a hierarchy of a root class directory and root class files, a class making procedure comprising an interface for receiving a class specification as well as for receiving a message (abstract, col. 2, lines 46-67, col. 3, lines 1-14, col. 4, lines 46-67 and col. 5, lines 1-28), setting the search path of hierarchical file system to the content of the class instance or class file under unchanged current directory (col. 9, lines 14-22). Densmore does not explicitly teach a level of indirection from application programming interfaces used by a class

locator, the wrapper indirection level providing for different caches to be used for the selected elements.

However, Angel teaches creating a Java byte code wrapper for locating or searching finding or retrieving the class from the constant pool of the class instance and for providing a level of indirection from the application programming interfaces (APIs) for each query objects (sections 0016, 0126, 0132 0135 , 0161 0168; also see fig. 17).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Densmore with the teachings of Angel, wherein the interface for receiving or selecting the element such as a class specification as input in the system provided therein (Densmore's col. 4, lines 52-59 and col. 5, lines 5-28), would incorporate the use of creating a wrapper for selected object (Angel's section 0016 and 0168), into the system of Densmore for the purpose of efficiently searching/locating the classes in the class path (Densmore's col. 9, lines 6-9 and lines 15-32).

Claim 6 is essentially the same as claim 5 except that it is directed to a computer readable medium rather than a method, and is rejected for the same reason as applied to the claim 5 hereinabove.

With respect to claims 7-9, Densmore teaches a method of locating classes as discussed in claim 5.

Densmore teaches creating a root class with a class hierarchy of objects in a hierarchical file system, objects are organized as class instances and classes, the data are contained in the class variables and/or the class instance variables, a hierarchy of a

root class directory and root class files, a class making procedure comprising an interface for receiving a class specification as well as for receiving a message (abstract, col. 2, lines 46-67, col. 3, lines 1-14, col. 4, lines 46-67 and col. 5, lines 1-28), setting the search path of hierarchical file system to the content of the class instance or class file under unchanged current directory (col. 9, lines 14-22). Densmore does not explicitly teach a zip file, Java classes and Java package manager.

However, Angel teaches Zip file under a virtual machine (VM) runtime system that interprets and runs byte code, Java byte code, JNI (sections 0131-0135).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Densmore with the teachings of Angel, wherein the interface for receiving or selecting the element such as a class specification as input in the system provided therein (Densmore's col. 4, lines 52-59 and col. 5, lines 5-28), would incorporate the use of zip file from the VM runtime machine for selected object (Angel's section 0131), into the system of Densmore for the purpose of efficiently searching/locating the classes in the class path (Densmore's col. 9, lines 6-9 and lines 15-32).

With respect to claim 10, Densmore discloses parsing the class path into names of elements (col. 7, lines 65-67 and col. 8, lines 1-26); determining which elements are viable for caching; and initiating creation of caches and wrappers (hierarchy of root class and class instance directory are a wrapper: col. col. 4, lines 60-67 and col. 5, lines 1-28) for those elements which are viable (see abstract, see figs: 1, 3 and 4, col. 5, lines 54-67 and col. 6, lines 1-45 and col. 7, lines 55-67 and col. 8, lines 1-26).

Densmore discloses creating a root class with a class hierarchy of objects in a hierarchical file system, objects are organized as class instances and classes, the data are contained in the class variables and/or the class instance variables, a hierarchy of a root class directory and root class files, a class making procedure comprising an interface for receiving a class specification as well as for receiving a message (abstract, col. 2, lines 46-67, col. 3, lines 1-14, col. 4, lines 46-67 and col. 5, lines 1-28), setting the search path of hierarchical file system to the content of the class instance or class file under unchanged current directory (col. 9, lines 14-22). Densmore does not explicitly teach initiating creation of wrappers for each selected elements and the wrapper providing a level of indirection from application programming interfaces used by class locator to search the classes.

However, Angel teaches creating a Java byte code wrapper for locating or searching finding or retrieving the class from the constant pool of the class instance and for providing a level of indirection from the application programming interfaces (APIs) for each query objects (sections 0016, 0126, 0132 0135 , 0161 0168; also see fig. 17).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Densmore with the teachings of Angel, wherein the interface for receiving or selecting the element such as a class specification as input in the system provided therein (Densmore's col. 4, lines 52-59 and col. 5, lines 5-28), would incorporate the use of creating a wrapper for selected object (Angel's section 0016 and 0168), into the system of Densmore for the

purpose of efficiently searching/locating the classes in the class path (Densmore's col. 9, lines 6-9 and lines 15-32).

With respect to claims 11-14, Densmore discloses wherein the viability of an element for caching is dependent on the ease of tracking which elements have had changes in them; wherein the viability of an element for caching is determined based on it being a predetermined type; checking a registry to see if the element already has a cache associated with it; and determining if an existing cache is up to date (col. 5, lines 54-67, col. 6, lines 1-67, col. 7, lines 1-67 and col. 8, lines 1-26).

With respect to claim 15, Densmore discloses means for receiving requests to search a multi-elements class path for classes (search path: col. 9, lines 6-9 and lines 15-32), means for transferring such request through a wrapper associated (hierarchy of root class and class instance directory are a wrapper: col. col. 4, lines 60-67 and col. 5, lines 1-28).

Densmore discloses creating a root class with a class hierarchy of objects in a hierarchical file system, objects are organized as class instances and classes, the data are contained in the class variables and/or the class instance variables, a hierarchy of a root class directory and root class files, a class making procedure comprising an interface for receiving a class specification as well as for receiving a message (abstract, col. 2, lines 46-67, col. 3, lines 1-14, col. 4, lines 46-67 and col. 5, lines 1-28), setting the search path of hierarchical file system to the content of the class instance or class file under unchanged current directory (col. 9, lines 14-22). Densmore does not explicitly teach a wrapper providing a level of indirection associated with each element to invoke

element specific search methods, the search methods using different caches for different elements.

Densmore does not explicitly teach a level of indirection from application programming interfaces used by a class locator, the wrapper indirection level providing for different caches to be used for the selected elements.

However, Angel teaches creating a Java byte code wrapper for locating or searching finding or retrieving the class from the constant pool of the class instance and for providing a level of indirection from the application programming interfaces (APIs) for each query objects (sections 0016, 0126, 0132 0135 , 0161 0168; also see fig. 17).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Densmore with the teachings of Angel, wherein the interface for receiving or selecting the element such as a class specification as input in the system provided therein (Densmore's col. 4, lines 52-59 and col. 5, lines 5-28), would incorporate the use of creating a wrapper for selected object (Angel's section 0016 and 0168), into the system of Densmore for the purpose of efficiently searching/locating the classes in the class path more efficient (Densmore's col. 9, lines 6-9 and lines 15-32).

With respect to claim 16, Densmore disclose at least one such element specific search method comprising a cache associated with such element (col. 9, lines 1-32 and col. 4, lines 60-67)

With respect to claim 17, Densmore discloses means for parsing the multi-elements class path into names of elements (col. 7, lines 65-67 and col. 8, lines 1-26);

means for determining whether each element is a variable cache candidate and for creating a cache for such variable candidates (col. 5, lines 54-67, col. 6, lines 1-67, col. 7, lines 1-67 and col. 8, lines 1-26) and means for creating indirection wrappers (hierarchy of root class and class instance directory are a wrapper: col. col. 4, lines 60-67 and col. 5, lines 1-28).

Densmore discloses creating a root class with a class hierarchy of objects in a hierarchical file system, objects are organized as class instances and classes, the data are contained in the class variables and/or the class instance variables, a hierarchy of a root class directory and root class files, a class making procedure comprising an interface for receiving a class specification as well as for receiving a message (abstract, col. 2, lines 46-67, col. 3, lines 1-14, col. 4, lines 46-67 and col. 5, lines 1-28), setting the search path of hierarchical file system to the content of the class instance or class file under unchanged current directory (col. 9, lines 14-22). Densmore does not explicitly teach means for creating indirection wrappers for each element to map class searches to each element for independent handling.

Densmore does not explicitly teach a level of indirection from application programming interfaces used by a class locator, the wrapper indirection level providing for different caches to be used for the selected elements.

However, Angel teaches creating a Java byte code wrapper for locating or searching finding or retrieving the class from the constant pool of the class instance and for providing a level of indirection from the application programming interfaces (APIs) for each query objects (sections 0016, 0126, 0132 0135 , 0161 0168; also see fig. 17).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Densmore with the teachings of Angel, wherein the interface for receiving or selecting the element such as a class specification as input in the system provided therein (Densmore's col. 4, lines 52-59 and col. 5, lines 5-28), would incorporate the use of creating a wrapper for selected object (Angel's section 0016 and 0168), into the system of Densmore for the purpose of efficiently searching/locating the classes in the class path (Densmore's col. 9, lines 6-9 and lines 15-32).

With respect to claim 18, Densmore discloses the cache for each viable candidate comprises a name of class (col. 5, lines 9-15 and lines 49-60).

With respect to claim 19, Densmore teaches a class path manager as discussed in claim 17.

Densmore teaches creating a root class with a class hierarchy of objects in a hierarchical file system, objects are organized as class instances and classes, the data are contained in the class variables and/or the class instance variables, a hierarchy of a root class directory and root class files, a class making procedure comprising an interface for receiving a class specification as well as for receiving a message (abstract, col. 2, lines 46-67, col. 3, lines 1-14, col. 4, lines 46-67 and col. 5, lines 1-28), setting the search path of hierarchical file system to the content of the class instance or class file under unchanged current directory (col. 9, lines 14-22). Densmore does not explicitly teach a zip file, Java classes and Java package manager.

However, Angel teaches Zip file under a virtual machine (VM) runtime system that interprets and runs byte code, Java byte code, JNI (sections 0131-0135).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Densmore with the teachings of Angel, wherein the interface for receiving or selecting the element such as a class specification as input in the system provided therein (Densmore's col. 4, lines 52-59 and col. 5, lines 5-28), would incorporate the use of zip file from the VM runtime machine for selected object (Angel's section 0131), into the system of Densmore for the purpose of efficiently searching/locating the classes in the class path (Densmore's col. 9, lines 6-9 and lines 15-32).

With respect to claims 20-21, Densmore discloses the directories are not caches and wherein the viability of an element for caching is dependent on the ease of tracking which elements have had changes in them (see abstract, see figs. 1, 3-4; col. 2, lines 46-65 and col. 5, lines 17-42).

With respect to claim 22, Densmore discloses a class path manager that receives requests for identification or enumeration of classes of classes in the class path; a cache for a cache viable element of the class path; a wrapper for such cache viable element that receives such request from the class path manager (abstract, col. 2, lines 46-67 and col. 3, lines 1-13; also see col. 5, lines 1-42; col. 7, lines 11-18 and col. 8, lines 6-15).

Densmore discloses creating a root class with a class hierarchy of objects in a hierarchical file system, objects are organized as class instances and classes, the data

are contained in the class variables and/or the class instance variables, a hierarchy of a root class directory and root class files, a class making procedure comprising an interface for receiving a class specification as well as for receiving a message (abstract, col. 2, lines 46-67, col. 3, lines 1-14, col. 4, lines 46-67 and col. 5, lines 1-28), setting the search path of hierarchical file system to the content of the class instance or class file under unchanged current directory (col. 9, lines 14-22). Densmore does not explicitly teach a wrapper for each such cache viable element that receives such requests from the class path manager and that provides a transparent level of indirection to services that are specific to such cache viable element.

Densmore does not explicitly teach a level of indirection from application programming interfaces used by a class locator, the wrapper indirection level providing for different caches to be used for the selected elements.

However, Angel teaches creating a Java byte code wrapper for locating or searching finding or retrieving the class from the constant pool of the class instance and for providing a level of indirection from the application programming interfaces (APIs) for each query objects (sections 0016, 0126, 0132 0135 , 0161 0168; also see fig. 17).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Densmore with the teachings of Angel, wherein the interface for receiving or selecting the element such as a class specification as input in the system provided therein (Densmore's col. 4, lines 52-59 and col. 5, lines 5-28), would incorporate the use of creating a wrapper for selected object (Angel's section 0016 and 0168), into the system of Densmore for the

purpose of efficiently searching/locating the classes in the class path (Densmore's col. 9, lines 6-9 and lines 15-32).

Claim 23 is essentially the same as claim 5 except that it is directed to a computer readable medium rather than a method, and is rejected for the same reason as applied to the claim 5 hereinabove.

5. Claim 24 is rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No 5,187,786 issued to Densmore et al. (hereinafter Densmore) in view of Pub. No.: US 2001/0047510 A1 of Angel et al. (hereinafter Angel), and further in view of US Patent No. 6,212,564 issued to Harter et al. (hereinafter Harter).

With respect to claim 24, Densmore in view of Angel discloses a computer-readable medium as discussed in claim 23.

Densmore and Angel disclose substantially the invention as claimed.

Densmore and Angel do not teach checking a data/time stamp on the element.

However, Harter discloses the current data/time storing in the cache (col. 3, lines 38-51 and col. 5, lines 1-15).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Densmore in view of Angel with the teachings of Harter by incorporating the use of current date/time storing in the cache (Harter's col. 3, lines 38-51), into the system of Densmore for the purpose of efficiently searching/locating the classes in the class path (Densmore's col. 9, lines 6-9 and lines 15-32).

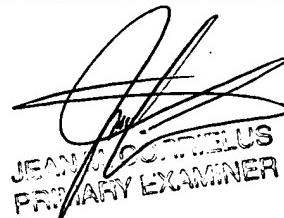
Contact Information

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Anh Ly whose telephone number is (571) 272-4039 or via E-Mail: ANH.LY@USPTO.GOV or fax to (571) 273-4039. The examiner can normally be reached on TUESDAY – THURSDAY from 8:30 AM – 3:30 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Breene, can be reached on (571) 272-4107 or **Primary Examiner Jean Corrielus (571) 272-4032.**

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Any response to this action should be mailed to: Commissioner of Patents and Trademarks, Washington, D.C. 20231, or faxed to: **Central Fax Center (571) 273-8300**



JEAN CORRIELUS
PRIMARY EXAMINER

ANH LY
MAR. 8th, 2006